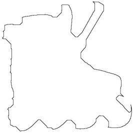
Reconstitution 3D à partir de photographies : Détection de contour

Extraction de la silhouette

Intersection des silhouettes

images/silhouettes.jpg

Choix des Types

Type : Flottant-4octet, Complex-8octet

- Plus rapide : processeur à opérations flottantes
- Pas de dépassement : exposant sur 8-bits $\longrightarrow$ max $= 10^{256}$

Structure :

Tableaux numpy à 5 dimensions :

1	2	3	4	5
dx	dy	x	y	i
index voisinage		index image		composantes

Complexité

Temporelle

En théorie :

- Opérations vectorielles : GPU. $\longrightarrow$ noté $\nu(n)$
- Autres : CPU.

En pratique :

- Opérations vectorielles : numpy, boucles en C.
- Autres : Python

Objectif : Vectoriser au maximum les opérations.

Spatiale

Vectorisation $\longrightarrow$ Complexité spatiale très grande :
$\theta(t^2 \cdot n^2) \approx (8\text{octet} \cdot 1000 \cdot 1\text{Mpix} = 8\text{Gib})$

Coût de copie : Relativement élevé $\approx \nu(n^2)$

Manipulation bas niveau du hardware (registres vectoriels du GPU) $\longrightarrow$ rapide

Procédé Général

1. Espace LAB
2. Filtrage fréquentiel : Texture
3. Filtrage spatial : Bordures
4. Lissage parabolique
5. Sommation des réponses
6. Seuillage
7. Opérations topologiques
8. Approximation polygonale

Espace LAB

$$L = \frac{R + G + B}{3}$$

$$A = \frac{G - R + 255}{2}$$

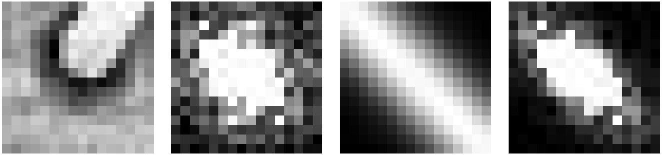
$$B = \frac{G - B + 255}{2}$$

Complexité : $\theta(\nu(n^2))$

1er Filtrage - Domaine fréquentiel - Texture

Procédé

Convolution dans le domaine spatial $\Longleftrightarrow$ **Multiplicatioin** dans le domaine fréquentiel



Sous-image - Transformée - Filtre Gaussien - Réponse

Sommation de tous les pixels $\longrightarrow$ valeur de texture.

Complexité spatiale : $\theta(t^2 \cdot n^2)$

1^{er} Filtrage - Domaine fréquentiel - Texture

Transformée de Fourier

Définition :

- Continue : $F(f)(s) = \frac{1}{\sqrt{2\pi}} \int_a^b e^{-2i\pi sx} \cdot f(x)\,\mathrm{d}x$
- Discrete : $FD(u)(k) = \frac{1}{\sqrt{N}} \sum_{n=0}^{n=N-1} e^{-2i\pi \frac{kn}{N}} \cdot un$

Propriété de convolution : $u \star v = DF^{-1}(DF(u) \cdot DF(v))$ (Au facteur près)

Complexité temporelle

- Sur un vecteur : $\theta(n \cdot \nu(n))$
- Sur une image : $\theta(n\nu(n^2))$

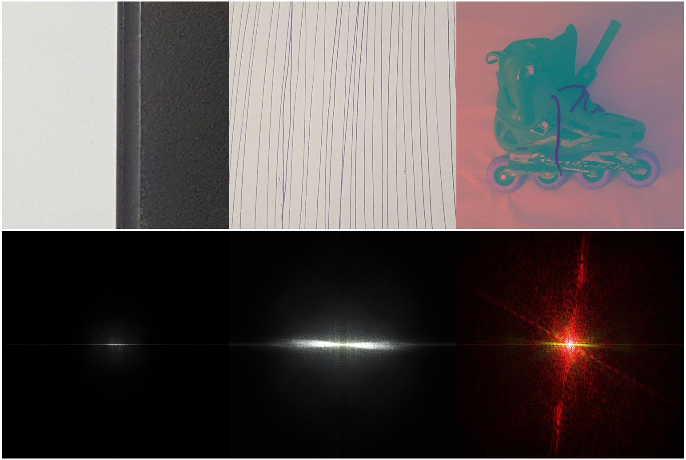
Complexité spatiale : En place

Transformée de Fourier rapide : Diviser pour régner

- Sur un vecteur : $\theta(\log_2(n)\nu(n))$
- Sur une image : $\theta(\log_2(n)\nu(n^2))$

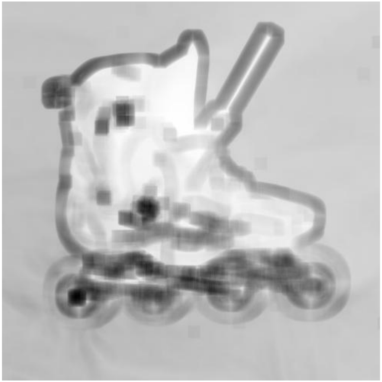
1^{er} Filtrage - Domaine fréquentiel - Texture

Exemples



1^{er} Filtrage - Domaine fréquentiel - Texture

Résultats



taille : 16, θ : $\frac{\pi}{2}$, σ : 4

$2^{\text{ème}}$ Filtrage - Domaine spatial - Bordures

Procédé

Pour chaque pixel : somme pondérée des pixels du voisinage.

Exemple : gradient simple

$$\begin{array}{|ccc|}\hline 35 & 42 & 27 \\ 11 & 12 & 14 \\ 0 & 7 & 4 \\ \hline\end{array} \quad \begin{array}{|ccc|}\hline 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \\ \hline\end{array} \quad \begin{array}{|ccc|}\hline 35 & 42 & 27 \\ 0 & 0 & 0 \\ 0 & -7 & -4 \\ \hline\end{array}$$

$\longrightarrow 93$

Complexité :

- Spatiale : En place
- Temporelle : $\theta(t^2\nu(n^2))$

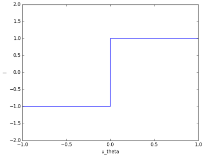
$2^{\text{ème}}$ Filtrage - Domaine spatial - Bordures

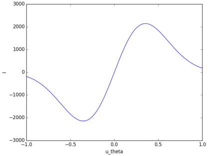
Filtre de Canny

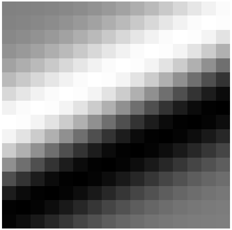
Filtre optimal pour les arêtes en "pas"

Dérivée d'une gaussienne :

$$h(x) = x \cdot e^{-\frac{x^2}{\sigma^2}}$$

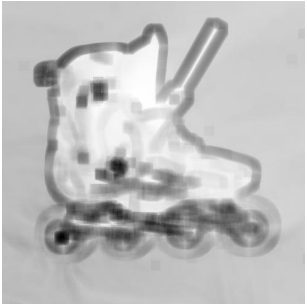


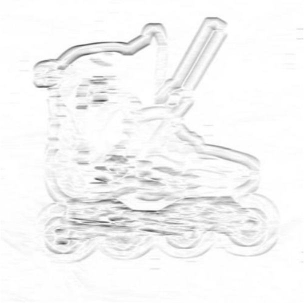




2ème Filtrage - Domaine spatial - Bordures

Résultats





$\theta : \frac{pi}{2}, \sigma : 1$

Lissage parabolique

Procédé

Approximation parabolique du voisinage : $ax^2 + bx + c$
Fonction lissée :

$$\frac{c}{\text{distance au max local}} = \frac{c^+}{\left(\frac{|b|}{2a^+}\right)}$$

Procédé :

	Opération	complexité
1	Génération du voisinage	selon hardware
2	Calcul de la coordonnée sur u_θ	$\theta(\nu(t^2))$
3	Etablissement du systeme	$\theta(t^2\nu(n^2))$
4	Pivot de gauss	$\theta(t^2\nu(n^2))$
5	Calcul de la fonction lissée	$\theta(\nu(n^2))$
	Total	$\theta(\nu(n^2))$

Lissage parabolique

Résultats

Sommation des réponses

Procédé

30 composantes :

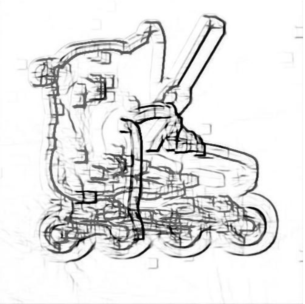
- 3 couleurs + 3 couleurs * 4 directions = 15 composantes avant filtrage
- pour chaque composante, filtrage par deux filtres de Canny d'écart type 1 et 2 = 30 composantes

Choix des coefficients de sommation ?

Approche optimale : apprentissage supervisé

Ici : choix empirique

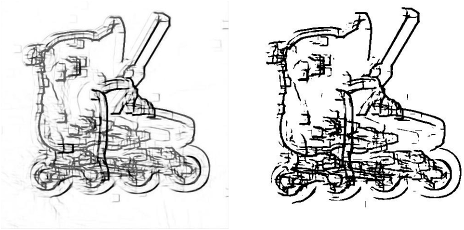
Résultats



Seuillage

Conversion en image binaire par seuillage :

$$\begin{cases} 1 & \text{si } I > 0 \\ 0 & \text{sinon} \end{cases} \tag{1}$$



Opérations binaires - Topologie

Définitions

Même procédé de filtrage mais sur des images binaires

Erosion

0	1	0
1	1	1
0	1	0

0	0	0	0	0
0	1	0	1	0
0	1	1	1	0
1	1	1	1	1
1	1	1	1	1

0	0	0	0	0
0	0	1	1	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	1

Dilatation

0	0	0
1	1	1
0	0	0

0	0	0	0	0
0	1	1	1	0
0	0	0	0	1
1	1	1	1	1
1	1	1	1	1

0	0	0	0	0
0	0	0	0	0
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0

Fermeture

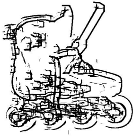
Dilatation + Erosion

Opérations binaires - Topologie

Résultats

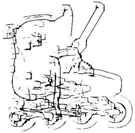
Erosion par :

0	1	0
1	1	1
0	1	0

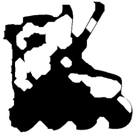


Fermeture par :

0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
0	0	0	0	0
0	0	0	0	0



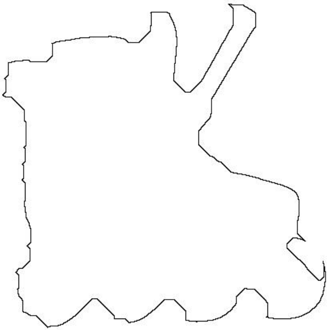
Auquel on applique des rotations dans 8 directions différentes



Approximation polygonale

Parcours du contour

Parcours dans le sens trigonométrique
Départ du pixel le plus en haut
Direction initiale : Bas
Priorités de déplacement : Gauche - Bas - Droite - Haut



Approximation polygonale

Approximation polygonale

Initialisation : Point le plus haut - Point le plus bas
Sur chaque segment, successivement :

- Calcul de D distance max du contour au segment
- Si $D >$ seuil, on ajoute le point le plus éloigné
- Lorsque pour tout segment $D <$ seuil, on sort de la boucle

Conclusion

Précision : moyenne
Temps : moyen $\approx$ 45s

Annexe - Transformée de Fourier rapide I

Notation :

- $N = 2p$
- $n = n_1 + pn_2 \qquad n_1 \in (0, p-1) \quad n_2 \in 0, 1$
- $k = 2k_1 + k_2 \qquad k_1 \in (0, p-1) \quad k_2 \in 0, 1$
- $a = e^{-\frac{2i\pi}{N}}$

Alors :

$$F(u)(k) = \sum_{n=0}^{N-1} u(n) \cdot a^{kn} \quad (2)$$

$$F(u)(2k_1 + k_2) = \sum_{n_1=0}^{1} \sum_{n_2=0}^{p-1} u(n_1 + pn_2) \cdot a^{k \cdot (n_1 + pn_2)} \quad (3)$$

$$= \sum_{n_1=0}^{1} \sum_{n_2=0}^{p-1} u(n_1 + pn_2) \cdot a^{kn_1} \cdot a^{kpn_2} \quad (4)$$

Annexe - Transformée de Fourier rapide II

De plus : $a^{kpn_2} = a^{2p \cdot k_1 n_2} \cdot a^{k_2 n_1 p} = a^{k_2 n_1 p}$

Ainsi : $F(u)(2k_1 + k_2) = \sum_{n_1=0}^{1} a^{kn_1} \cdot \sum_{n_2=0}^{p-1} u(n_1 + pn_2) \cdot a^{k_2 n_1 p}$

Posons : $v(n_1, k_2) = \sum_{n_2=0}^{p-1} u(n_1 + pn_2) \cdot a^{k_2 n_1 p}$

Finalement : $F(u)(2k_1 + k_2) = v(0, k_2) + a^k \cdot v(1, k_2)$

Le calcul de v est celui d'une transformée de Fourier discrète.

On applique récursivement la décomposition